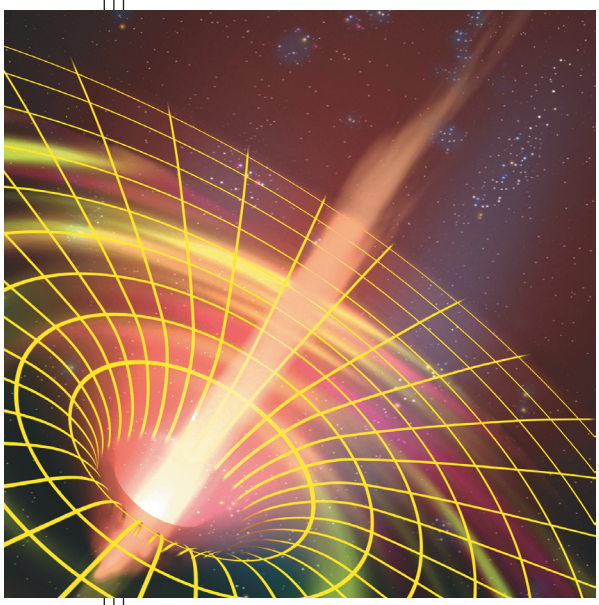


ENERGY-EFFICIENT COMPUTING FOR EXTREME-SCALE SCIENCE

David Donofrio, Leonid Oliker, John Shalf, and Michael F. Wehner, *Lawrence Berkeley National Laboratory*
Chris Rowen, *Tensilica*
Jens Krueger, *Fraunhofer Institute, Germany*
Shoaib Kamil and Marghoob Mohiyuddin, *University of California, Berkeley*

A many-core processor design for high-performance systems draws from embedded computing's low-power architectures and design processes, providing a radical alternative to cluster solutions.

The computational power required to accurately model extreme problem spaces, such as climate change, requires more than a business-as-usual approach. Building ever-larger clusters of commercial off-the-shelf (COTS) hardware will be increasingly constrained by power and cooling—with power consumption projected to be hundreds of megawatts for exascale-class problems according to recent DARPA and DOE reports. It makes more sense therefore to leverage the considerable innovation of the low-power architectures developed for embedded computing markets and design a machine capable of the exaflops performance (1 billion-billion floating-point operations per second) required for this and similarly demanding scientific applications.

To that end, we have developed Green Flash, an application-driven design that combines a many-core processor with novel alternatives to cache coherence and autotuning to improve the kernels' computational efficiency. This approach can achieve two-orders-of magnitude improvement in computational efficiency for climate simulation relative to a conventional symmetric multiprocessor (SMP) approach.

The challenge of moving high-performance computing architecture toward exaflops has staggering economic and political ramifications. The computational power required for extreme-scale modeling accurate enough to inform critical policy decisions requires a new breed of extreme-scale computers. The "A Page from Embedded Computing" sidebar describes the architectural philosophy behind Green Flash.

To test our design philosophy, we chose a truly exascale problem: kilometer-scale models of the global atmosphere system requiring simulations 1,000 times faster than real time. The kilometer-scale model decomposes Earth's atmosphere into 20 billion individual cells, demanding a machine with unprecedented performance.

Applying energy-efficient, embedded processors, although a crucial first step, is not in and of itself sufficient to meet this challenge. The computing industry has arrived at a rare inflection point: Fundamental principles of computer architecture are open to question, and new ideas are being explored. Green Flash not only offers a glimpse of how design processes that have been successful in the embedded space can be applied to scientific

 Published by the IEEE Computer Society

A PAGE FROM EMBEDDED COMPUTING

Peter Ungaro, CEO of Cray Computing, recently remarked that "Our current technologies can get us to the 10-20 petaflops range. But then to start to think about 100 [petaflops], we really need a major shift in technology."[1]

The high-performance computing community has long subscribed to architectural specialization as the best way to boost efficiency, but design and verification costs and lead times have made the cost of creating full-custom designs impractical. Thus, to think about exaflops other than in the context of science fiction means abandoning the idea of building ever-larger clusters and turning to another set of proven design strategies that don't come at a prohibitive cost.

In our search for a radical alternative, we turned to the embedded-processor market, which successfully addresses the custom and cost issues. The industry relies on sophisticated tool chains that enable the rapid and cost-effective turnaround of power-efficient semicustom design implementations appropriate to each application.

Our design, Green Flash, leverages the same tool chains to design power-efficient exascale systems, tailoring embedded chips to target scientific applications. Rather than ask, What kind of scientific applications can run on our high-performance computing cluster? after it arrives, we have turned the question around to ask, What kind of system should be built to meet the needs of the most important science problems? This approach lets us realize the most substantial gains in energy efficiency because we essentially peel back the complexity of a high-frequency microprocessor design point to reduce waste—wasted opcodes, wasted bandwidth, waste caused by orienting architectures toward serial performance. We also change the notion of commodity from that of component-level integration of clusters to integration of commodity circuit designs within a chip for a system-on-chip.

By using hardware-software cotuning, our design enables rapid hardware design and establishes a feedback path from application programmer to hardware designer. By combining an autotuning environment for software optimization with an emulation platform based on an FPGA, we can simultaneously develop software optimizations and a semispecialized processor design. Essentially, we have not only built on proven ideas, but we have taken them in a new direction.

Reference

1. G. Huang, "Cray's Comeback: CEO Peter Ungaro on Clouds, Exaflops, and the Future of Supercomputing," 30 July 2009; www.xconomy.com/seattle/2009/07/30/crays-comeback-ceo-peter-ungaro-on-clouds-exaflops-and-the-future-of-supercomputing.

computing, but also addresses some of the most daunting problems of managing the exponential growth of on-chip-parallelism across the entire information technology (IT) industry.

MODELING EARTH'S CLIMATE SYSTEM

Current-generation climate models are comprehensive representations of the systems that determine Earth's climate. Models prepared for the *IPCC Fourth Assessment Report: Climate Change 2007*[1] coupled submodels of the atmosphere, ocean, and sea ice to provide simulations of the past, present, and future climate. Models already being prepared for the next report will represent the major remaining climate system components—the terrestrial and oceanic biosphere, the Greenland and Antarctic ice sheets, and certain aspects of atmospheric chemistry.

Each subsystem model has its own strengths and weaknesses and introduces a particular amount of uncertainty into climate projections. Current computational resources limit the resolution of these submodels, thereby contributing to the uncertainty. Resolution constraints on atmospheric process models, for example, do not allow clouds to be resolved, which means that model developers must rely on subgrid-scale parameterizations that are based on statistical methods. Simulations with these constraints produce cloud distributions that do not correlate well with observations.

Such disagreements can be traced to cumulus convection parameterization. Current global atmospheric models have resolutions of approximately 200 km—many times larger than individual clouds. A few groups are attempting to develop models at the limit of cumulus parameterization validity (approximately 25 km), but the necessary century-scale integrations are just barely feasible on the largest current computing platforms. Although this increase in horizontal fidelity should rectify many issues, the fundamental limitations of cumulus parameterization are likely to remain.

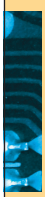
For this reason, it makes more sense to simulate cloud processes directly rather than model them statistically. At a horizontal grid spacing of approximately 1 km, a model could resolve cloud systems individually, providing a direct numerical simulation. However, because numerical stability requirements impose time-step limitations, the computational burden of fluid dynamics algorithms scales nonlinearly with the number of grid points. Consequently, the resources to carry out century-scale simulations of Earth's climate would overwhelm the capability of any traditional machine.

Resource requirements

In previous work,[2] we estimated the resources needed for a resolution of approximately 1 km. To fine-tune these estimates, we have since partnered with David Randall's group at Colorado State University (CSU) that is using a mesh representation of the globe with an icosahedron as the starting point. By successively bisecting the sides of the triangles making up this object, the group was able to generate a remarkably uniform mesh on the sphere. However, this is not the only way to discretize the globe at this resolution; a variety of independent resolving models are necessary to make credible projections about climate

change. Consequently, we wanted to be sure that Green Flash could run a class of global climate models, not just a particular model.

We originally estimated 10 petaflops as the sustained computational rate necessary to simulate Earth's climate 1,000 times faster than it actually occurs. An updated estimate of the requirements for the CSU model raised that to as high as 70 petaflops—an example of the considerable uncertainty in making these estimates. As the CSU model matures, we expect to determine this rate even more accurately. An exaflops-scale machine would provide multiple realizations of individual simulations, a necessary tool in addressing the climate system's statistical complexities. The exact peak flops rate required would depend greatly on the machine's potential efficiency.



To help maintain backward and general-purpose compatibility, the processor's instruction set architecture is expandable but must be functional enough to allow general-purpose code execution.

A strawman decomposition

Without sufficient parallelism in the climate problem, these enormous sustained computational rates are not even imaginable. Fortunately, the CSU group has demonstrated that the icosahedral formulation of cloud-system resolving models at the kilometer scale can offer plenty of opportunity to decompose the physical domain. Their decomposition bisects the triangles composing the icosahedron 12 successive times, producing a global mesh with 167,772,162 vertices spaced 1 to 2 km apart. It is then possible to apply a logically rectangular two-dimensional domain-decomposition strategy horizontally to the icosahedral grid. Choosing square segments of the mesh containing 64 grid points each (8 × 8) results in 2,621,440 horizontal domains. The vertical dimension offers additional parallelism. Assuming that we could decompose 128 layers into eight separate vertical domains, the total number of physical subdomains could be 20,971,520.

Even given 20-million-way parallelism, we continued to pursue the strawman decomposition, keeping in mind the practical constraints on an SMP core's performance. With a single core assigned to each subdomain, individual cores must be capable of a computational rate of about 3.5 gigaflops for the icosahedral code to achieve a simulation 1,000 times faster than real time. These rates are based on the computational efficiency rates of current mainstream rates. The efficiency gained through autotuning can bring the peak flops rate requirement down considerably. About 25 Mbytes of memory would also be required per subdomain as would about 7,500 nearest-neighbor messages per second with a size of 8 to 10 Kbytes each. This last requirement translates to a bandwidth of about 78 Mbytes per second between nearest-neighbor processors.

Designing 128 cores onto a single chip would result in 163,840 individual sockets—numbers that were not implausible. We were thus encouraged to take our strawman decomposition to the design stage.

DESIGN PROCESS

Because power constraints have long directed the development of embedded architectures, we began with an embedded core and some of the sophisticated tool chains developed to minimize time from architectural specifications to the application-specific IC. We then looked at how we could maximize efficiency by tuning the hardware and software to optimize performance as well as how we could provide rapid design prototypes and cope with fault resilience.

Leveraging an embedded tool chain

The sophisticated tool chains for developing the system-on-chip application-specific ICs popular in the embedded computing market give designers the flexibility to combine verified functional units in myriad ways to rapidly produce semicustom designs. To test these ideas, we adopted the tool chain from Tensilica,[3] an embedded-design firm. The chain starts with a base architecture, to which a designer can add floating-point support to a processor or perhaps choose a larger cache or local store. Adding features to the processor core (or removing them) is as simple as clicking a checkbox or selecting from a dropdown menu.

The tool then selects the unit from its library and integrates it into the design—which substantially reduces the writing and rewriting of the full custom logic typically required when changing a processor's architecture. To help maintain backward and general-purpose compatibility, the processor's instruction set architecture is expandable but must be functional enough to allow general-purpose code execution. The tools also allow designers to flexibly define application-specific extensions to the base instruction set architecture. Of course, the tools have their limits (a designer can't have hundreds of read ports from a single memory, for example), but their flexibility vastly outweighs any inherent restrictions.

Much like current high-performance computing designs, our approach continues to use off-the-shelf components except at a finer grain. Rather than using entire off-the-shelf processors at a socket-level granularity, we can tailor individual functional units within a core and their interconnections to create a semicustom system-on-chip (SOC) design.

The ability to rapidly generate processor cores that are tailored to scientific applications makes these tools compelling, but the excessive overhead in verifying hardware and creating a usable software stack for each new processor negates any time saved in hardware development. To address this drawback, the tools generate optimizing compilers—test benches as well as a functional simulator—in parallel with the design's register transfer logic. Constructing the processor with verified building blocks and automatically generating test benches greatly reduce the risk and time spent in formal verification.

Rapid design prototyping

Traditionally, the complexity of coding in Verilog or VHDL versus C++ or Python and the inability to emulate large designs have outweighed the speed and accuracy advantages of using field-programmable gate arrays (FPGAs). However, FPGA use has become much more practical over the past decade because, unlike commercial microprocessors, FPGAs are not experiencing a clock-rate and power plateau. The lookup table count on FPGAs continues to increase, enabling the emulation of more complex designs. In addition, FPGA clock rates have been growing steadily, closing the gap between emulated and production clock rates. Recent advances in FPGA I/O features have made accessing large, dynamic memories much more palatable.

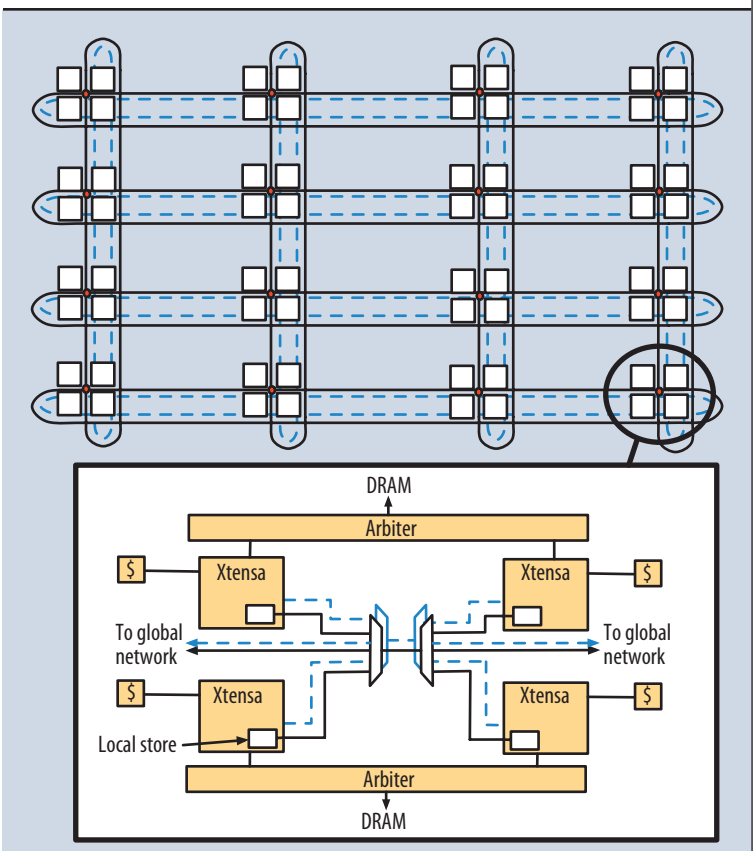


Figure 1. The on-chip network fabric for the Green Flash system-on-chip. A concentrated torus network fabric yields the highest performance and most power-efficient design for scientific codes.

To accelerate the creation of prototype system designs, we are using the Research Accelerator for Multiple Processors (RAMP),[6] an FPGA emulation platform that makes the hardware configuration available for evaluation while the actual hardware is still on the drawing board. RAMP is a cooperative effort among six universities to build a new standard emulation system for parallel processors.

Although the steady growth in FPGA lookup table count has enabled the emulation of more complex designs, a strawman architecture of 128 cores per socket requires emulating more than the two or four cores that will fit on a single FPGA. To address this limitation, we have employed version 3 of the Berkeley Emulation Engine (BEE3), a board populated with four Virtex-5 155 FPGAs, each with two dedicated channels of double data rate memory, connected in a ring with a crossover connection.

Using the BEE3, we effectively emulate eight networked cores, each running at 33 MHz. To scale beyond eight cores, the BEE3 includes 10-Gbit Ethernet connections, allowing the boards to be linked and enabling the emulation of an entire socket. There is significant precedent for emulating massively multithreaded architectures across multiple FPGAs. The Berkeley RAMP Blue project, for example, demonstrated the emulation of more than 1,000 cores using a stack of 16 BEE2 boards.[7]

Maximizing efficiency

Opting to follow the design philosophy that the best way to reduce power consumption and increase efficiency is to reduce waste, we chose an architecture with a very simple in-order core and no branch prediction. Because the climate model's demands for memory and communication are high, both aspects drive Green Flash's core design. Reducing the computational burden through autotuning also contributes to efficiency. Finally, hardware-software cotuning tunes the hardware to the autotuned software for additional efficiency gains.

Network topology. Our experience evaluating the STI Cell processor[4] shows that, for memory-intensive applications, cores with a local store use a higher percentage of the available dynamic RAM (DRAM) bandwidth. On the basis of these results, we decided to include a local store in our processor architecture. As Figure 1 shows, the design uses a torus network fabric with two on-chip networks. Predictably, most of the communication among the climate model's subdomains is nearest neighbor. We did

PHOTONIC NETWORKS: A MORE EFFICIENT NETWORK INTERCONNECT

Power efficiency requires reducing the power consumption of all system components. With these highly efficient tiny processing elements there is a danger that communication bottlenecks—both in energy and time—will result in a less efficient overall system. To mitigate this danger, long-term research requires exploring interconnect architectures that will both increase performance and reduce energy use.

One promising approach is to combine 3D CMOS integration with research into silicon photonics to build hybrid electronic-photonic interconnects on-chip.[1,2] Designers place photonic detectors and emitters along with specialized low-power photonic switching elements on a special interconnect layer and interface them with processing elements using conventional electronic routers. Figure A shows how the switching elements work. Large-scale communications occur over photonic links, which have several strong advantages over electronic networks. Energy consumption for photonics is less dependent on signaling rate and distance compared to electronics, and the photonic switches are much simpler as they do not require buffers or repeaters.

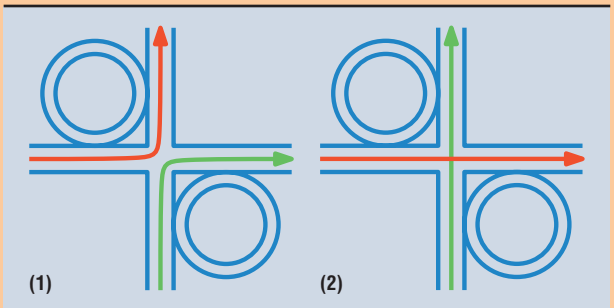


Figure A. Photonic switching elements. (1) Light is coupled onto a perpendicular path; (2) messages propagate straight through. The lack of distance and complex structures are strong advantages over a purely electrical interconnect.

Preliminary research with messaging patterns arising from scientific applications shows that such hybrid networks have the potential to bring major gains in efficiency, due to their lower power consumption combined with fast propagation speed. Early research studies done in collaboration with the Lightwave Research Laboratory at Columbia University, for example, show that a hybrid electronic-photonic interconnect composed of ring resonators can deliver 27x better energy efficiency than electrical interconnects alone.[3]

References

1. C. Batten et al., "Building Many-Core Processor-to-DRAM Networks with Monolithic CMOS Silicon Photonics," *IEEE Micro*, Special Issue: Micro's Top Picks from Hot Interconnects 16, vol. 29, no. 4, 2008, pp. 8-21.
2. A. Shacham, K. Bergman, and L.P. Carloni, "Photonic Networks-on-Chip for Future Generations of Chip Multiprocessors," *IEEE Trans. Computers*, vol. 57, no. 9, 2008, pp. 1246-1260.
3. G. Hendry et al., "Analysis of Photonic Networks for a Chip Multiprocessor Using Scientific Applications," *Proc. 2009 3rd ACM/IEEE Int'l Symp. Networks-on-Chip* (NOCs 09), IEEE CS Press, 2009, pp. 104-113.

additional experiments with cycle-accurate models of an on-chip packet-switched network to determine that a concentrated torus topology provides superior performance and energy efficiency for codes in which a nearest-neighbor communication pattern dominates.[5] We are currently targeting a core with a clock speed of 500 MHz, a 32-Kbyte conventional error correction code (ECC)-protected cache per core, and a 128-Kbyte local store. The availability of a conventional cache will allow code to be incrementally ported to use the local store. Each socket of 128 cores will have a 50-Gbyte-per-second interface to DRAM.

The traditional cache-coherent memory consistency schemes typical of most modern SMPs make fine-grained synchronization among cores very difficult, and greatly increase the amount of undesired interprocessor data movement. For example, to achieve our target execution rate on 20 million processors, we must compute on a local mesh size that is 8 × 8 × 10 cells. We have observed that the code would spend 90 percent of its time in communication if it were to run on a conventional cache-based hardware, due to the overhead penalty of exchanging extremely small messages between cores.

In Green Flash, we have added specialized hardware to each core to enable extremely low-overhead messaging between cores, bringing the communication overhead below 20 percent of the total execution time. We have used Tensilica's tools to create multiple designer-defined ports with a simple first-in, first-out interface, and each port can send and receive a word-sized data packet on each clock. This ultra-low-overhead streaming interface bypasses the cache to minimize latency and connects to one of the on-chip torus networks. The narrow network is for address exchange; the wider torus network is for bulk data exchange using asynchronous direct memory access (DMA) data transfers. The address space for each processor's local store is mapped into the global address space, and the data exchange is done as a DMA from local store to local store.

From a logical programming view, all processors are directly connected to each other, but physically are connected using a concentrated torus network to the chip's 2D planar geometry. To further simplify programming, a traditional cache hierarchy is also in place to allow the slow porting of codes to the more efficient interprocessor network. To minimize power, we are investigating the use of photonic interconnects for the intercore network, which could prove to be an efficient way of transferring long messages. The "Photonic Networks: A More Efficient

Network Interconnect" sidebar describes the advantages of this approach.

Autotuning. Communication was not our only challenge in the climate model computation. Meeting the requirement of simulating at 1000× real time per core in a power-efficient design is a daunting task, so to optimize the code and reduce the computational burden, we created an autotuning framework that automatically searches a range of optimizations to improve the application kernels' computational efficiency. The autotuner first systematically applies compiler optimizations and then uses domain-specific knowledge of the algorithm to take more aggressive steps, such as loop reordering, to produce optimal, but functionally equivalent, code. In this way, it maintains performance across a diverse set of architectures.

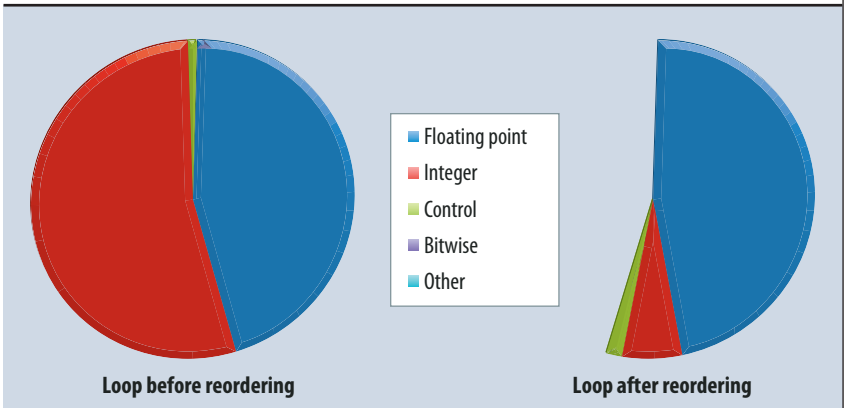


Figure 2. Effect of optimization on a single loop in the climate model. In addition to greatly reducing the instruction count, optimization reduced the cache footprint of this loop by more than 100 times. With software tuning, Green Flash can reduce a per-core computational requirement of 3.5 gigaflops to a more feasible 0.5 gigaflops.

Figure 2 shows the results for the climate model. We ran the autotuning framework using the Tensilica architectural simulator, reducing the cache footprint and overall instruction count and increasing the kernel's computational density. We first generated the original requirement of 3.5 gigaflops per core using a machine that ran with approximately 5 percent efficiency. Autotuners, combined with hardware optimizations, will play a key role in dramatically increasing the efficiency of Green Flash. Through these combined optimizations, we expect Green Flash to realize a two-orders-of-magnitude increase in efficiency.

Hardware-software codesign. Conventional approaches to hardware design use benchmark codes to search for a power-efficient architecture. However, modern compilers fail to generate even close to optimal code for target machines, which strongly implies that a benchmark-based approach to hardware design does not exploit the full performance potential of the architecture design points and can lead to possibly suboptimal hardware solutions. The success of autotuners proves the feasibility of generating efficient code using domain knowledge. Therefore, we created cotuning as a technique to tailor the hardware to autotuned software to get better energy efficiency. Using our autotuning technology, we can automate the exploration for the optimal combination of tuned software *and* hardware in a coordinated design cycle.

As Figure 3 shows, our cotuning approach incorporates extensive software tuning into the hardware design process. The autotuned software tailors the application to the hardware design point under consideration by empirically searching software implementations to find the best mapping of software to microarchitecture.

As a demonstration of our proposed cotuning methodology, we used the Smart Memories multiprocessor (based on Tensilica cores) as the target architecture and three widely used kernels from scientific computing—dense matrix-matrix multiplication, stencil codes, and sparse matrix vector multiplication. As part of exploring the hardware design space, we varied four hardware parameters: number of cores, whether caches are hardware- or software-managed, cache size per core, and total memory bandwidth available.

We used tools to estimate the area and power of each hardware configuration that had the corresponding best software configuration, which we obtained through autotuning. As Figure 4 shows, power and area efficiencies improved dramatically for the three kernels.

One hindrance to practical cotuning is the large hardware-software design space to be explored to tailor the hardware design parameters to the target applications. Conventional hardware design approaches use a software simulation of the hardware to perform this exploration. However, cotuning in Green Flash must explore the software design space at each hardware design point, making it impractical to cotune using software simulation.

Instead, we took advantage of the Tensilica tool chain's ability to create synthesizable register-transfer logic for any processor and, by loading this design onto an FPGA, we were able to emulate a potential processor design running 500 times faster than a functional simulator. With this speedup, designers can benchmark true applications rather than having to rely on representative code snippets or statically defined benchmarks. More important, this speed advantage does not come at the expense of accuracy; FPGA emulation is arguably much more accurate than a software simulation environment because it truly represents the hardware design.

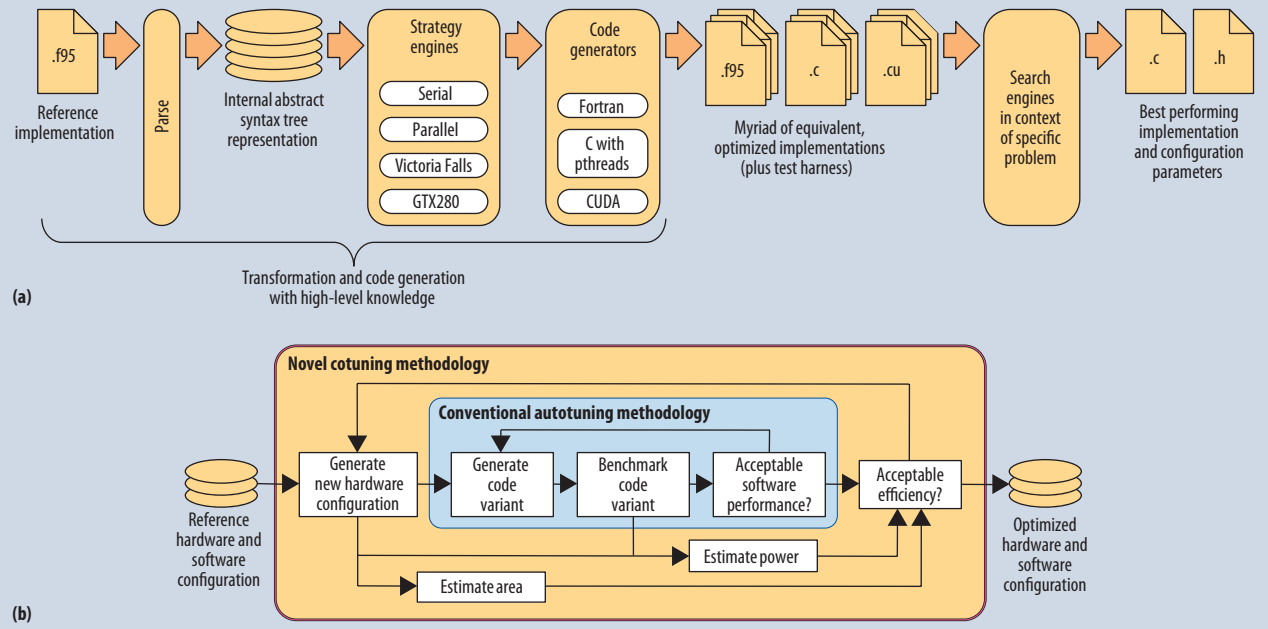


Figure 3. Cotuning in the Green Flash design. (a) Conventional autotuning uses source code generators and search heuristics to empirically choose an efficient software implementation given a high-level representation of a kernel. (b) Hardware-software cotuning extends conventional hardware design space exploration by using autotuning to tailor software to each hardware design point.

The hardware-software codesign process enables scientific application developers to directly participate in the design process for future supercomputers in an unprecedented way. With this fast, accurate emulation environment, designers can run and benchmark the actual climate model as it is being developed and use cotuning to quickly search a large design space.

We believe that these experiments outline a path for bringing the concept of hardware-software codesign—already prevalent in embedded design practices—into the realm of supercomputing system design.

SCALING UP

This article focuses primarily on hardware and software design methodology. However, in considering any system of this scale, a myriad of system software issues come to the forefront, such as scalable operating systems, fault resilience infrastructure, and the development of entirely new programming models to make billion-way parallelism more tractable.

Fault resilience

An important question arises when proposing a 20-million-processor computing system: How do you deal with fault resilience? Although the problem is certainly not trivial, neither is it unusual. As long as the total number of discrete chips is not dramatically different, any large-scale design faces the challenge of aggregating conventional server chips into large-scale systems.

Across silicon design processes with the same design rules, hard failure rates are proportional to the number of system sockets and typically stem from a mechanical failure. Soft error rates are proportional to the chip surface area—not how many cores are on a chip—and bit error rates tend to increase with clock rate. The Green Flash architecture is unremarkable in all these respects and should not pose challenges beyond those that a conventional approach faces.

To deal with hard errors, designers often add redundant cores per chip to cover defects. An old trick in the memory business, the strategy is apparent in designs such as the 188-core Cisco Metro chip, and it is entirely feasible for our design as well. Moreover, Green Flash's low power dissipation per chip (7 to 15W) will reduce the mechanical and thermal stresses that often result in a hard error.

To address soft errors, we have included all the basics for reliability and error recovery in the memory subsystem, including full ECC protection for all hierarchical levels. Green Flash's low target clock frequency provides a lower signal-to-noise ratio for on-chip data transfers. Finally, to enable faster rollback if an error does occur, our design makes it possible to incorporate a nonvolatile RAM controller onto each SMP so that each node can perform a local rollback as needed. This strategy

enables much faster rollback, relative to user-space checkpointing.

The Blue Gene system at Lawrence Livermore National Laboratory uses similar fault resilience strategies and contains a comparable number of sockets to Green Flash, yet its mean time between failures (MTBF) is 7 to 10 days[8]—much longer than systems with far fewer processor cores. Because we tailor our architecture to the application, Green Flash can deliver more performance than a machine with a comparable number of sockets, thus reducing its exposure to both hard and soft errors. It proves that carefully applying well-known fault-resilience techniques together with a few novel mechanisms that extend fault resilience, such as localized nonvolatile RAM checkpoints, can yield an acceptable MTBF for extreme-scale implementations.

Programming model

Future hardware constraints and growth in explicit on-chip parallelism will likely require a mass migration to new algorithms and software architecture that is as broad and disruptive as the migration from vector to parallel computing systems that occurred 15 years ago. Applications and algorithms will need to rely increasingly on fine-grained parallelism and strong scaling and support fault resilience.

History shows that the application-driven approach we are using for Green Flash offers the most productive strategy for evaluating and selecting among the myriad choices for refactoring algorithms for full scientific application codes as we move through this transitional phase. We are exploring novel programming models together with hardware support to express fine-grained parallelism to achieve performance, productivity, and correctness for leading-edge application codes in the face of massive parallelism and increasingly hierarchical hardware. The goal of this development thrust is to create a new software model that can provide a stable platform for software development for the next decade and beyond for all scales of scientific computing.

We have developed direct hardware support for both the message passing interface (MPI) and partitioned global address space (PGAS) programming models to enable scaling of these familiar single program, multiple data (SPMD) programming styles to much larger-scale systems. The modest hardware support enables relatively well-known programming paradigms to utilize massive on-chip concurrency and to use hierarchical parallelism to enable use of larger messages for interchip communication. The icosahedral formulation of the climate problem can expose a massive degree of parallelism through domain decomposition, which can use a 20-million processor computing system. The autotuning framework is rapidly evolving into a generalized code generator, which allows the programmer to express the solver kernels at a much higher level of abstraction—enabling a productive programming environment that supports portability, performance, and correctness without exposing scientists to the details of the computer architecture. We think this approach can be scaled out to support a broad range of codes that have such inherent explicit parallelism.

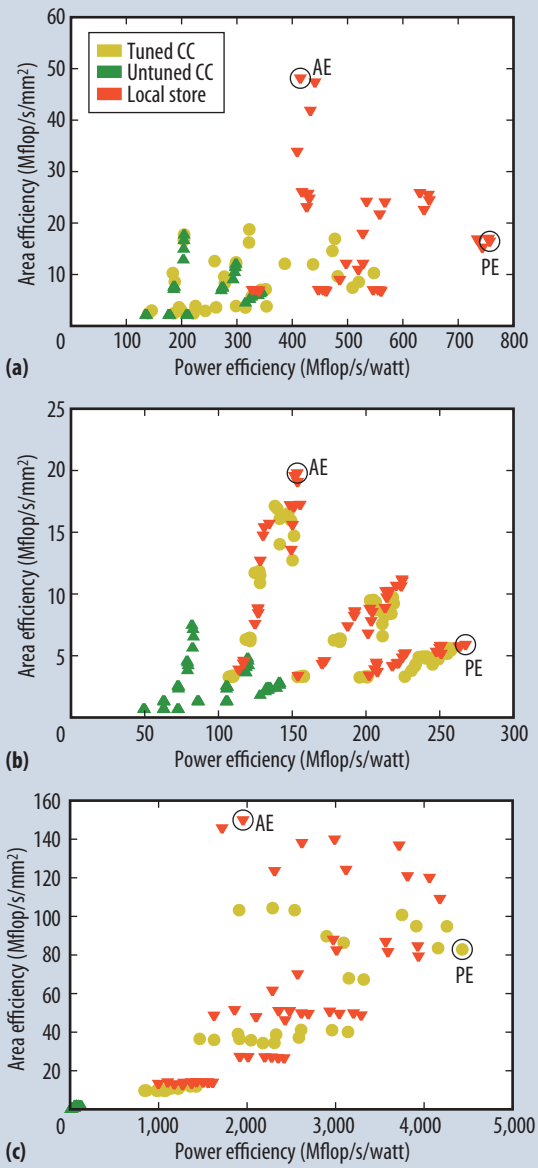


Figure 4. The advantages of cotuning for three kernel types common in scientific applications. AE and PE points denote configurations with highest area and power efficiencies. Improvements varied from 2x to 50x.

However, not all applications will be able to express parallelism through simple divide-and-conquer problem partitioning. We are only just beginning to explore new asymmetric and asynchronous approaches to achieving strong-scaling performance improvements from explicit parallelism. Techniques that resemble class static dataflow

methods are garnering renewed interest because of their ability to flexibly schedule work and to accommodate state migration to correct load imbalances and failures.

In the case of the climate code, we can use dataflow techniques to concurrently schedule the physics computations with the dynamic core of the climate code, thereby doubling our concurrency without moving to a finer domain decomposition. This approach also benefits from the unique interprocessor communication interfaces developed for Green Flash. Successful demonstration of the new parallelization procedure for a range of leading extreme-scale applications can then be utilized by other similar codes, accelerating development efforts for the entire field.

What's next?

Designs that follow our approach have the potential to open a market demand for massively concurrent components that can also be the building blocks for mid- and extreme-scale computing systems. New programming models must be part of a new software development ecosystem that spans all system scales so that the industry has a viable migration path from development to large-scale production computing systems. We have demonstrated the value of FPGA-based hardware emulation platforms, such as RAMP, in prototyping and running hardware prototypes at near-real-time speeds before they are built. Such a capability will make it possible to test full-fledged application code and advanced software development many years ahead of the hardware platform construction.

Although machines such as Blue Gene or SciCortex have demonstrated the advantages of using simple, low-power embedded cores, our approach goes beyond these traditional designs by optimizing data movement through explicit message queues and software controlled memories. Relative to models such as CUDA[9] (Compute Unified Device Architecture) and Streaming,[10] our simple hardware support for lightweight on-chip interprocessor synchronization and communication provides a straightforward approach to programming a massive array of processors. Rather than limit implementation to off-the-shelf embedded ASIC tools, we also investigated more exotic technologies, such as silicon photonic interconnects.

Cost is and will continue to be a critical driver in evolving new technologies. The scientific computing community cannot sustain the end-to-end cost of developing and maintaining technologies that apply only to the narrow market of leading-edge high-performance computing systems. Broad-based market support is a prerequisite to make such an ecosystem both practical and sustainable. We believe that our decision to draw from the embedded computing industry will produce technology that reduces economic and manufacturing barriers to constructing computing systems useful to science. It will also ensure that selected technologies have broad market impact for everything from the smallest handheld to the largest supercomputer. The investment will thus be the center of a sustainable software-hardware universe supported by applications across the IT industry.

For the past decade, the current methodologies of message-passing interfaces and Fortran have adequately served the development of high-performance computing applications. But parallelism is no longer an exotic problem. It is an industry-wide challenge that affects everything from cell phones to data centers. Future hardware constraints and growth in explicit on-chip parallelism will require a mass migration to new algorithms and software architecture—a migration as broad and disruptive as that from vector to parallel computing systems.

Green Flash represents a radical approach that breaks through the slow pace of incremental change. It demonstrates that application-driven computing design can foster a sustainable hardware-software ecosystem with broad-based support across the IT industry. In evolving Green Flash, we explored practical advanced programming models together with lightweight hardware support mechanisms that allow programmers to use massive on-chip concurrency.

Green Flash has provided insights into how designers can evolve massively parallel chip architectures through a feedback path that closely couples application, algorithm, and hardware design. Application-driven design ensures that hardware design is not driven by reactions to hardware constraints—reactions that ignore programmability and delivered application performance. Our exploration of the climate model allowed us to investigate questions that cut across all application areas and have ramifications for the next generation of fully general-purpose architectures. Ultimately, we envision an architecture that can exploit reusable components from the mass embedded computing market while improving programmability for a many-core design. The future building blocks of a high-performance computing system will serve the performance and programmability needs of the smallest high-performance, energy-efficient embedded system all the way to extreme-scale machines. C

Acknowledgments

We thank Mark Horowitz and the rest of the Smart Memories Team of Stanford University for early support and advice. We thank the Berkeley Wireless Research Center for early and ongoing assistance with the RAMP platform. We thank Dave Randall's modeling group in the Department of Atmospheric Science at Colorado State University for early access to their icosahedral model. Finally, we would like to acknowledge the Berkeley ParLab and the View from Berkeley discussion that

led us to take this direction for HPC architecture. All authors from LBNL were supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231.

References

1. Intergovernmental Panel on Climate Change (IPCC); www.ipcc.ch/publications_and_data/publications_and_data_reports.htm.
2. M. Wehner, L. Oliker, and J. Shalf, "Towards Ultra-High Resolution Models of Climate and Weather," *Int'l J. High Performance Computing Applications*, Apr. 2008, pp. 149-165.
3. Tensilica Inc., "Hardware and Software Development Tools;" www.tensilica.com/products/hw-sw-dev-tools.htm.
4. S. Williams et al., "Scientific Computing Kernels on the Cell Processor," *Int'l J. Parallel Programming*, vol. 35, no. 3, 2007, pp. 263-298.
5. G. Hendry et al., "Analysis of Photonic Networks for a Chip Multiprocessor Using Scientific Applications," *Proc. Int'l Symp. Networks-on-Chip* (NOCs 09), 2009; www.cs.columbia.edu/~luca/research/pnocs_NOCS09.pdf.
6. Research Accelerator for Multiple Processors (RAMP); http://ramp.eecs.berkeley.edu.
7. D. Burke et al., "RAMP Blue: Implementation of a Manycore 1008 Processor FPGA System," *Proc. Reconfigurable Systems Summer Institute 2008* (RSSI 08), July 2008; www.rssi2008.org/proceedings/papers/presentations/19_Burke.pdf.
8. "Into the Wide Blue Yonder with Blue Gene/L," *Science and Technology Rev.*, Lawrence Livermore National Laboratory, Apr. 2005; www.llnl.gov/str/April05/Seager.html.
9. Nvidia Inc.; www.nvidia.com/object/cuda_home.html.
10. AMD/ATI Inc., "ATI Stream Software Development Kit;" http://developer.amd.com/gpu/ATIStreamSDK.

David Donofrio *is a computer systems engineer at Lawrence Berkeley National Laboratory and a member of the Science Driven System Architecture Group at the National Energy Research Supercomputing Center. His research interests include computer architecture and multicore performance optimization. Donofrio received a BS in computer engineering from Virginia Tech. He is a member of the IEEE Computer Society. Contact him at ddonofrio@lbl.gov.*

Leonid Oliker *is a staff computer scientist in the Future Technologies Group at Lawrence Berkeley National Laboratory. His research interests include supercomputing evaluation, multicore autotuning, and power-efficient computing. Oliker received a PhD in computer science from the University of Colorado at Boulder. Contact him at loliker@lbl.gov.*

John Shalf *is a staff computer scientist at Lawrence Berkeley National Laboratory and the Group Leader of the Science Driven System Architecture Group of the National Energy Research Supercomputing Center and leads the Green Flash Project. His research interests include computer architecture, programming models, and frameworks for large-scale scientific application development. Shalf received a degree in electrical engineering from Virginia Tech. He is a member of the IEEE Computer Society. Contact him at jshalf@lbl.gov.*

Michael F. Wehner *is a member of the Scientific Computing Group at the Lawrence Berkeley National Laboratory. His research interests include the design of global climate models and the analysis of their output. Wehner received a PhD in nuclear engineering from the University of Wisconsin-Madison. Contact him at mfwehner@lbl.gov.*

Chris Rowen *is the CTO and founder of Tensilica. His research interests include computer architecture, parallel programming models, and energy-efficient hardware architecture. Rowen received a PhD in electrical engineering from Stanford University. Contact him at rowen@tensilica.com.*

Jens Krueger *is a research assistant at the Fraunhofer Institute for Industrial Mathematics and a visiting researcher at the Lawrence Berkeley National Laboratory Future Technologies Group. His research interests include high-performance and scientific computing focused on future hardware architectures and code optimization. Contact him at jtkrueger@lbl.gov.*

Shoaib Kamil *is a PhD student in the Department of Computer Science at the University of California, Berkeley. His research, conducted with the Future Technologies Group at Lawrence Berkeley National Laboratory, is in autotuning, power efficiency, and interconnect optimization. Contact him at skamil@eecs.berkeley.edu.*

Marghoob Mohiyuddin *is a PhD student in the Department of Electrical Engineering and Computer Science at the University of California, Berkeley, and a member of the Future Technologies Group at Lawrence Berkeley National Laboratory. His research interests include computer architecture and performance tuning for scientific computing. Contact him at marghoob@eecs.berkeley.edu.*